Digital Filtering and Sampling Fundamentals

Terry S. Yoo
Office of High Performance Computing and Communications
The National Library of Medicine, National Institutes of Health

1 Introduction

Digital sampling theory is intrinsic to computer graphics, image processing, and visualization. Representing complex shapes and flows using pixels, voxels, and other discrete atomic data types creates repercussions that pervade all of volume graphics. Familiar artifacts such as aliasing can only be understood through digital sampling and filtering theory. Likewise, effective reconstruction, the inverse of sampling, requires a clear understanding of digital sampling theory.

This chapter is a review or introduction (depending on your background) of the digital theories that underlie sampling and filtering. This material will be covered from a practical viewpoint: what should we know about the basic principles of digital sampling theory? What are the repercussions of which we should be aware? How do these theoretical results affect our output images? What can you do about it, and where are the pitfalls that should be avoided? This presentation is intended as a cursory introduction or review only. We will not explore proofs or rigorous mathematics associated with the presented theorems or tools.

For example, the presentation of the Fourier transform will be far from exhaustive, not even listing all of the important mathematical properties of the transform, but rather just those properties that are relevant to volume graphics. We also will not be presenting any implementations. However, we will provide pointers to references for more rigorous treatments of this material, as well as forums, journals, and venues where current research may be found.

2 Images

In this course, we are principally concerned with volume images. A volume image, ϕ, is a function or mapping from $\Re^3$ to $\Re^n$ where n = 1 for the typical, scalar–valued volume. More precisely:

$$\phi : U \mapsto \Re \text{ and } U \subset \Re^3, \tag{1}$$

where U is the domain of the volume. The image ϕ will also frequently be written as a function $\phi(x, y, z)$.

Throughout this introductory chapter, however, many example images will be presented either as projections from a 1-D or a 2-D domain. Most exercises in sampling theory are more easily examined in 1D (at least in print), and generalized to higher dimensions. Wherever we explore images of fewer than three dimensions, we will use the function notation (*i.e.*, $\phi(x)$ or $\phi(x, y)$).

We distinguish discrete, sampled digital images from the description of images as contiuous mappings of functions in real space. If F is a discrete sampling of ϕ then we can say that

$$F_{i,j,k} = \phi(x_i, y_i, z_i). \tag{2}$$

We also use notation to distinguish between discrete and continous differential operators. To take the approximate partial derivative in the x direction of a discretely sampled image we can say that

$$\phi_x(x_i, y_i, z_i) = \left.\frac{\partial \phi}{\partial x}\right|_{x_i, y_i, z_i} \approx \delta_x f_{i,j,k}, \tag{3}$$

where

$$\delta_x f_{i,j,k} \equiv \frac{f_{i+1,j,k} - f_{i-1,j,k}}{x_{i+1} - x_{i-1}} = \frac{f_{i+1,j,k} - f_{i-1,j,k}}{2h}, \tag{4}$$

where h is the grid spacing, which we normally assume to be 1–pixel width. The value $\delta_x f_{i,j,k}$ is an approximation of the instantaneous partial derivative of $\phi(x_i, y_i, z_i)$. taken at (x_i, y_i, z_i) in the x direction. Equation 4 is the method of *central differences* commonly used to take derivatives of uniformly, discretely sampled datasets.

3 Linear Filtering

When was the last time you used the convolution integral? If you've run a blurring function, taken a derivative, done some edge enhancement, or other simple operation, it is likely that you've used convolution (either the continuous integral form or the discrete iterative summation). Filters are used to make measurements, detect features, smooth noisy signals, and deconvolve operations from previous operations (e.g., deblurring the known optical artifacts from a telescope). The effect of the filter depends not on the operation of convolution, but rather on the nature of input filter kernel.

3.1 Convolution

Convolution, denoted with the operator $\otimes$ can be described in 1-D as a continuous operation applying a filter kernel $h(x)$ (itself an image) to an image $\phi(x)$ using the following integral form:

$$\phi(x) \otimes h(x) = \int_{-\infty}^{\infty} \phi(x - \tau) h(\tau)\, d\tau \tag{5}$$

Note that the expression $\phi(x) \otimes h(x)$ itself describes an image mapping. Thus the following equality holds:

$$\phi(x) \otimes h(x) = (\phi \otimes h)(x) \tag{6}$$

The expressions $\phi(x) \otimes h(x)$ and $(\phi \otimes h)(x)$ are used interchangably, with the position index x often omitted to help streamline and clarify the notation especially when higher dimensional (*e.g.*, 3-D) images are involved. In the practice of digital image processing, $h(x)$ typically does not have infinite extent, nor is it infinitely dense; however, when working in the continuous domain of digital filtering theory, functions or signals that can be considered infinitely wide (thus avoiding the complications of truncation) are often more convenient. We will elaborate later on discrete, bandlimited functions in the discussion on sampling.

We can generalize the 1-D convolution operation to 2-D and 3-D images. Thus, in 2-D, convolution becomes:

$$\phi(x, y) \otimes h(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \phi(x - \tau, y - \nu) h(\tau, \nu)\, d\nu\, d\tau \tag{7}$$

Similarly, convolution in 3-D is expressed as:

$$\phi(x, y, z) \otimes h(x, y, z) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \phi(x - \tau, y - \nu, z - \omega) h(\tau, \nu, \omega)\, d\omega\, d\nu\, d\tau \tag{8}$$

3.1.1 Example - Convolution as Noise Reduction (1-D)

One of the most common uses of convolution is as a filter for the suppression of noise in an image. Consider the Gaussian function as a smoothing filter kernel. If convolved with a relatively noisy signal, the result is an output image that locally averages the image intensities. The output image typically has reduced noise at the cost of some "sharpness" of the original image. That is, the resulting image has less perceived noise, but also has blurred features.

Figure 3.1.1 shows a 1-D step discontinuity, or edge. The figure ranges continuously from $x = 0$ to $x = 255$ with a step function at $x = 128$. The signal to noise ratio is approximately 4:1.

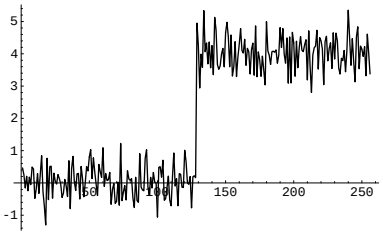
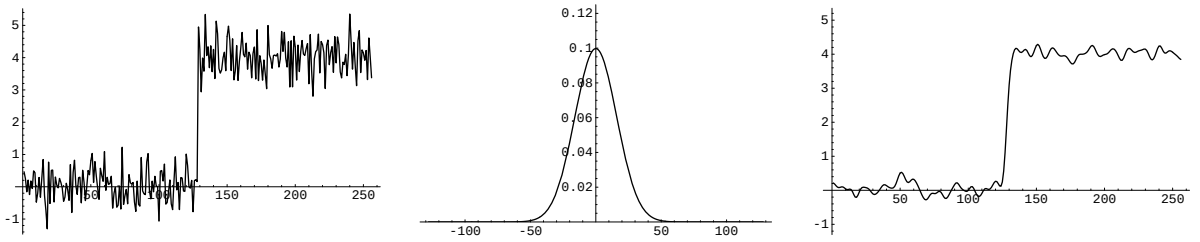


Figure 1: A 1-D Noisy Step Edge: An example input signal $\phi(x)$.



(a) Input $\phi(x)$

(b) a 1-D Gausssian kernel, $g(x, \sigma)$ where $\sigma = 16$

(c) output of $\phi(x) \otimes g(x, 16)$

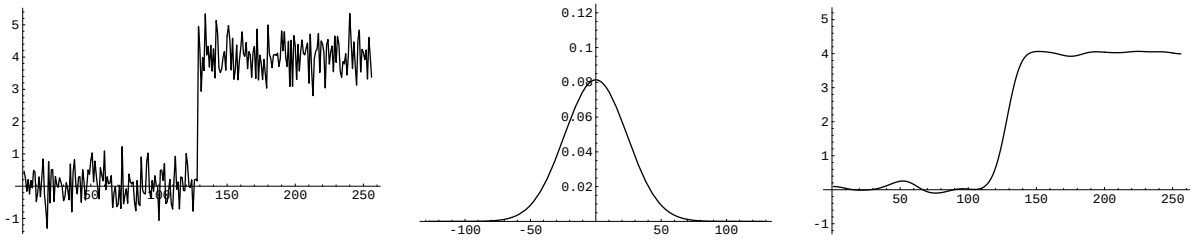
Figure 2: Gausssian filtering of input $\phi(x)$ where $\sigma = 16$

Figures 2, 3, and 4, show a progression of filtered versions of the input from figure 3.1.1 with Gaussian filter kernels of increasing aperture or width. Notice how the perceived noise in the signal decreases with aperture and the consequent softening of the gradient information at the discontinuity. This trade of detail and resolution for noise reduction is one of the many considerations in the design of linear filter systems.

3.2 Properties of the Convolution Operation

The properties of a linear filtering operation often depend on the nature of the filter kernel. For instance, the "shape" of the kernel will determine whether the operation remains invariant with respect to rotation. However, there are many inherent properties of the convolution operation. The convolution operation has many useful properties. It is linear, commutative, associative, and distributive over addition.

In mathematical terms, convolution is commutative and reflexive:

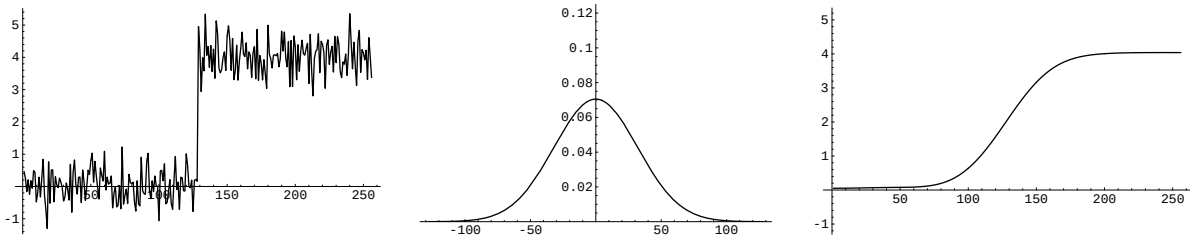


(a) Input $\phi(x)$

(b) a 1-D Gausssian kernel, $g(x, \sigma)$ where $\sigma = 24$

(c) output of $\phi(x) \otimes g(x, 24)$

Figure 3: Gausssian filtering of input $\phi(x)$ where $\sigma = 24$



(a) Input $\phi(x)$

(b) a 1-D Gausssian kernel, $g(x, \sigma)$ where $\sigma = 32$

(c) output of $\phi(x) \otimes g(x, 32)$

Figure 4: Gausssian filtering of input $\phi(x)$ where $\sigma = 32$

$$p \otimes q = q \otimes p \tag{9}$$

Convolution is also associative:

$$(p \otimes q) \otimes r = p \otimes (q \otimes r) \tag{10}$$

Convolution is distributive over addition:

$$p \otimes (q + r) = p \otimes q + p \otimes r \tag{11}$$

These combined properties create the justification for using convolution as the principal operation in linear filtering.

3.2.1 2-D Example - Convolution as Feature Detection and Enhancement

Linear filter kernels can be selected to detect and enhance features. For example, if a difference of two Gaussian functions is used as a filter kernel, edge information can be both detected and enhanced. The difference of an image filtered with two Gaussians of different apertures is a process called "unsharp masking." The result is an *edge-enhanced* image.

Specifically, given a 2-D input image generating function $\phi(x, y)$ and two Gaussian filter kernels of differing aperture, $g(x, y, \sigma_1)$ and $g(x, y, \sigma_2)$ where $\sigma_1 < \sigma_2$, an edge enhanced image $\phi\prime(x, y)$ can be formed by a linear combination of the two filters. That is:

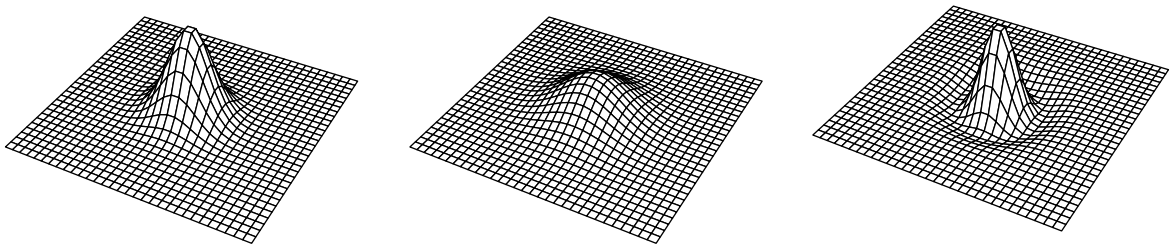
$$\phi'(x, y) = \phi(x, y) \otimes (g(x, y, \sigma_1) - g(x, y, \sigma_2)) \tag{12}$$

From equation 11 it follows that $\phi\prime(x, y)$ is also the subtraction of two filtered images.

$$\phi'(x, y) = (\phi(x, y) \otimes (g(x, y, \sigma_1)) - (\phi(x, y) \otimes g(x, y, \sigma_2))) \tag{13}$$

In other words, the difference of two "blurred" versions of the same image yields a version of the same image where edge information is accentuated.

A specific use of this technique is the detection of edges. If one applies two filter kernels that have unit gain (they do not amplify the signal) of differing widths and subtracts the results, the result is an image where regions of continuous intensity have a zero value, areas near boundaries are strongly negative or

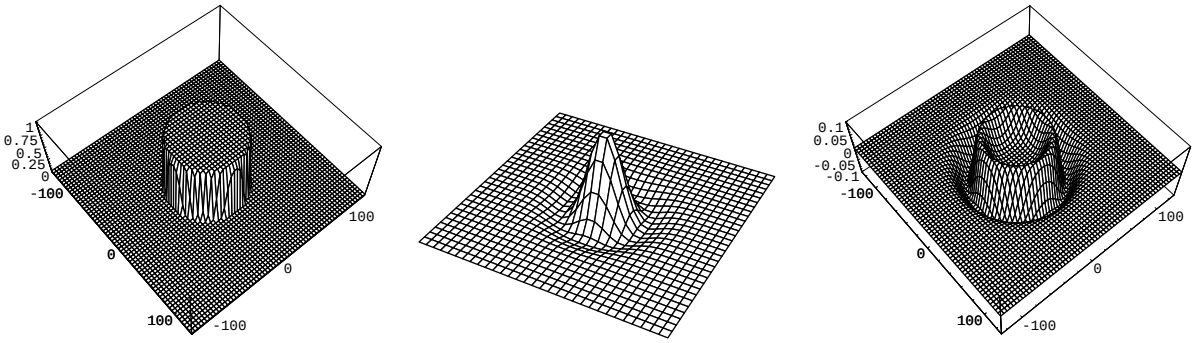


(a) 2-D Gaussian kernel $g(x, y, \sigma_1)$ with $\sigma_1 = 8$

(b) 2-D Gaussian kernel $g(x, y, \sigma_2)$ with $\sigma_2 = 12$

(c) $g(x, y, \sigma_1) - g(x, y, \sigma_2)$ where $\sigma_1 = 8$ and $\sigma_2 = 12$

Figure 5: a Difference of Gausssians filter kernel (depicted in 3-D as a height field).

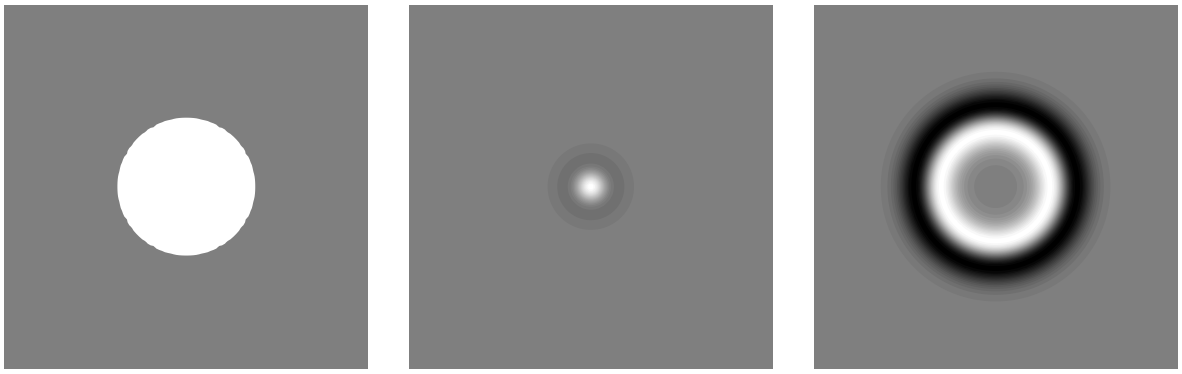


(a) 2-D Input $\phi(x, y)$ with $\sigma_1 = 16$

(b) $g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32)$

(c) $\phi(x, y) \otimes (g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32))$

Figure 6: a 2-D image filtered by a Difference of Gausssians (depicted in 3-D as a height field).



(a) 2-D Input $\phi(x, y)$ with $\sigma_1 = 16$

(b) $g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32)$

(c) $\phi(x, y) \otimes (g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32))$

Figure 7: a 2-D image filtered by a Difference of Gausssians (depicted as a 2-D greyscale density field).

(a) 2-D Input $\phi(x, y)$

(b) $2g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32)$

(c) $\phi(x, y) \otimes (g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32))$

Figure 8: a Difference of Gausssians filter used for contrast enhancement (depicted as a 2-D greyscale density field).

strongly positive (depending on which side of a boundary you are on) and the boundaries itself denoted by the closed curve of *zero-crossings*, those points where the function crosses from positive to negative values. Figure 5 shows such a differnce of Gaussian kernel as a 3-D projection where intensity is plotted as a height field above an x-y plane. Figure 6 is the same kernel show in figure 5 applied to a circular pulse function. These views are also represented as a 3-D projection of intensity as a height field. Figure 7 is the same functions as in figure 6 depicted as a 2-D intensity field. The areas of contiguous intensity result in a flat signal value of approximately zero. At the boundaries, there is a rise in the output intensity followed by a negative dip in intensity values. The zero crossing represents the boundary between the foreground intensity and the background.

With a slight adjustment of the filter kernel, applying coefficients to each of the Gaussian kernels, we can modify the edge detection kernel into a contrast enhancement kernel. Figure **??** shows an adjusted filter kernel with a combined gain of 1; that is, the difference of Gaussians will return the same value in areas of even intensity, but activates near boundaries to create a band of intensities that exaggerate the discontinuity.

3.3 Differentiation - Measurement Through Linear Filtering

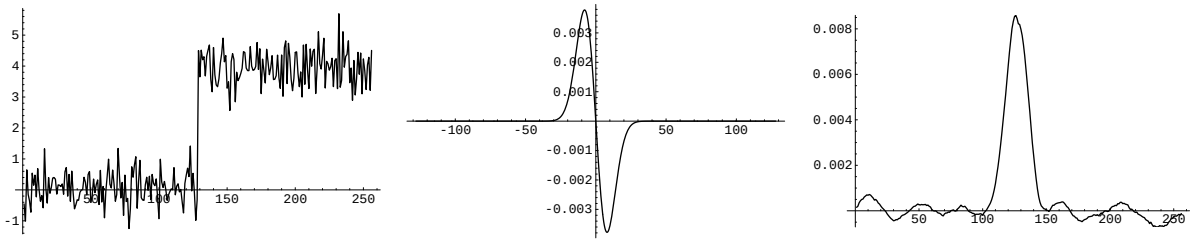
Differentiation is seldom portrayed as a convolution; however, the measurement of the instantaneous rate of change of intensities is in fact subject to all of the attributes of the convolution operation. The derivative operator is a convolution kernel that is infinitesimal in width, but is applied everywhere. Put another way, we can explicitly denote differentiation as convolution using the $\otimes$ operator as seen in the following example using the partial derivative in the x direction:

$$\frac{\partial}{\partial x}\phi = \frac{\partial}{\partial x} \otimes \phi \tag{14}$$

Differentiation is susceptible to noise, so it is often necessary to regularize a noisy dataset before computing its derivative. Therefore, derivatives are often taken of smoothed functions. It follows from the associative and commutative properties of convolution that:

$$h(x, y, z) \otimes \frac{\partial}{\partial x}\phi(x, y, z) = \frac{\partial}{\partial x}h(x, y, z) \otimes \phi(x, y, z) \tag{15}$$

In other words, the derivative of a function ϕ convolved by a filter kernel h is equivalent to convolving the



(a) 1-D Input $\phi(x)$ (b) $\frac{\partial}{\partial x}g(x,\sigma)$ where $\sigma = 3$ (c) $\phi(x) \otimes \frac{\partial}{\partial x}g(x,\sigma)$

Figure 9: Taking derivatives of a noisy input by convolution with the derivative of a filter kernel.

derivative of the h with ϕ. This suggests that one of the easiest ways to compute the derivative of a volume dataset is through convolution with the derivative of some smoothing kernel.

3.3.1 Example - Convolution as Measurement (1-D)

To show how derivatives can be approximated through convolution, we suggest the use of the Gaussian $g(x)$ as a regularizing (smoothing) filter kernel. Its infinitely differentiable properties make it attractive as a measurement and scaling function.

Figure 9 depicts a noisy 1-D input signal for which derivative information is desired. When convolved with the derivative of a Gaussian, the resulting output reports the derivative of a smoothed function. By the commutative and associative properties of convolution, it is also a smoothed version of the derivative of the input. This techniques for differentiating difficult functions can be extended to higher orders and is quite commonplace in the practice of volume image processing.

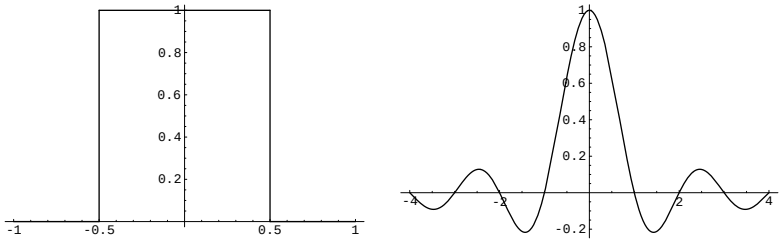
3.4 Convolution of Discretely Sampled Data

Convolution of discretely sampled data is similar to the continuous form, except that discrete summation is substituted for integration. Also, since discrete filter kernels seldom have infinite extent, the limits of the summations do not range from $-\infty$ to ∞ but are rather constrained to the size of the filter kernel. In 2-D, a discrete convolution looks like

$$P_{x,y} \otimes Q_{x,y} = \sum_{i}^{domain_x[Q]} \sum_{j}^{domain_y[Q]} P_{x-i,y-j} Q_{i,j} \tag{16}$$

The discrete version of convolution shares all of the attributes of its continous cousin. Convolution of discretely sampled data is commutative, associative, and distributive over addtion. Among its many uses, convolution can be used to smooth noisy data, detect or enhance edgeness, or make measurements of the image. Derivatives of sampled digital images can be approximated through discrete convolution using a well-chosen kernel. An example of discrete convolution for differentiation is the *central differences operation* shown in equation 4. From the perspective of linear filtering of discrete data, all such operations can now be seen to be either convolutions or adaptations of a convolution operation. For more information on differentiation of discrete image data through convolution, see the attached paper.

The primary caveat (as in *caveat emptor*) regarding discrete convolution is that all such discrete operations are subject to error induced by the effects of sampling, a class of artifacts known as *aliasing*. How much aliasing? We'll explore that later. Before we can appreciate the problems associated with sampling, we must first understand the nature of the sampling operation and its effects on the information embedded within the image. For that, we use an image analysis and manipulation tool called the Fourier Transform.



(a) The nearest neighbor interpolant (Box)

(b) Frequency Space nearest neighbor interpolant (Sinc)

Figure 10: The Box Filter and its Fourier Transform, the Sinc Function.

4 The Fourier Transform

One of the most important tools for understanding images (1-D, 2-D, 3-D or even higher) is the Fourier Transform. The Fourier transform of an image decomposes the input signal into a series sum of sinusoidal functions. The representation of an image as a series of frequencies is a computationally useful means of viewing and manipulating image data. The relationship between an input function and its Fourier transform is governed by the following equation:

$$\mathcal{F}(\phi(x)) = \int_{-\infty}^{\infty} \phi(x) e^{-i2\pi x\nu} dx = \Phi(\nu) \tag{17}$$

This relationship maps the domain and range of space, x and intensity, $\phi(x)$ to a new space where the domain and range are frequency, ν, and magnitude + phase, $\Phi(\nu)$, respectively. The transform space is often called *frequency space*, and the transformed input is described as the frequency space representation. As a transform, the process is also reversible. The inverse Fourier transform is:

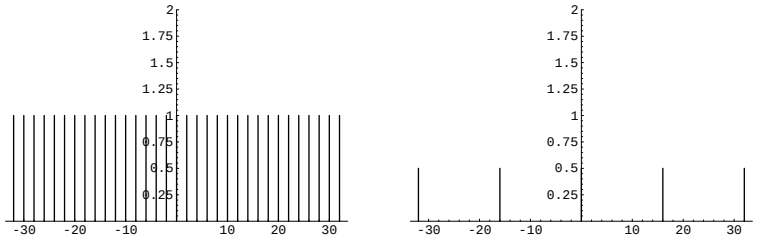
$$\mathcal{F}^{-1}(\Phi(\nu)) = \int_{-\infty}^{\infty} \Phi(\nu) e^{i2\pi x\nu} d\nu = \phi(x) \tag{18}$$

There are some simple generalizations of these formulas to 2-D, 3-D, and higher dimensions. Among its most important attributes, the Fourier transform is separable in each of the orthonnormal basis dimensions. One simply recasts the transform equation with additional independent variables of integration and within the integrand to extend it to higher dimensions. A complete description and exhaustive derivation of the Fourier transform and its properties is beyond the scope of this tutorial. However, there are some key properties of the frequency space and the transform that creates it that we will discuss to illuminate the problems of sampling in computer graphics.

4.1 Some Properties of the Fourier Transforms

Lets begin by looking at some functions commonly used in manipulating images. The four functions of interest as part of this discussion are the box filter (a square pulse, used as a nearest neighbor interpolant in reconstruction), the comb function (used to discretely sample data in the image domain), the pyramid function (a pyramid shaped filter used for linear interpolation in reconstruction), and the Gaussian (a good all purpose filter, except for its infinite extent).

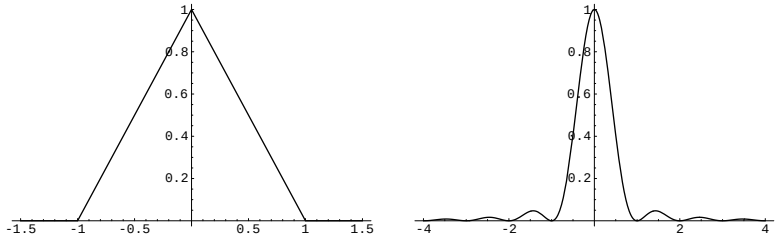
Figures 10, 11, 12, and 13 show the box, comb, pyramid, and Gaussian functions represented in image space along with their Fourier transforms, represented in frequency space. By studying this array of functions, one pair of important characteristics of the relationships between image space and freqency space become clear.



(a) Sampling Comb

(b) Frequency Space Sampling Function

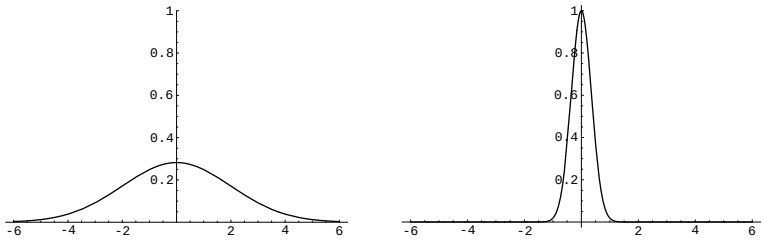
Figure 11: The Sampling Function and its Fourier transform.



(a) Linear Interpolating Pyramid Function

(b) The Fourier Transform of the Pyramid

Figure 12: The Linear Interpolation Function and its Fourier transform.



(a) Image Space Gaussian

(b) Frequency Space Gaussian

Figure 13: The Gaussian Function and its Fourier transform.

Spatial Scaling $\mapsto$ Inverse Frequency Scale Change An increase in the magnitude (height) of the function in the spatial domain leads to an inversely proportional broadening of the function in the frequency domain as well as an inversely proportional decrease of the height of the function in the frequency domain.

Frequency Scaling $\mapsto$ Inverse Spatial Scale Change The converse of the consequences of spatial scaling are true for frequency scaling. An increase in the magnitude (height) of the function in the frequency domain leads to an inversely proportional broadening of the function in the spatial domain as well as an inversely proportional decrease of the height of the function in the spatial domain.

Mathematically speaking:

$$\mathcal{F}(\phi(ax)) = \frac{1}{|a|}\Phi\left(\frac{\nu}{a}\right) \tag{19}$$

and

$$\mathcal{F}^{-1}(\Phi(a\nu)) = \frac{1}{|a|}\phi\left(\frac{\nu}{a}\right) \tag{20}$$

Conversely:

$$\mathcal{F}\frac{1}{|a|}\phi\left(\frac{\nu}{a}\right)(\phi(ax)) = \Phi(a\nu) \tag{21}$$

and

$$\mathcal{F}^{-1}\frac{1}{|a|}\Phi\left(\frac{\nu}{a}\right) = \phi(ax) \tag{22}$$

There are many other properties of the Fourier transform associated with its symmetry and its use of complex numbers to describe complicated behaviors of signals. We have neither the space or time to explore them all (For a more complete treatment, see the excellent text by E. O. Brigham.). However, the two related properties of spatial/frequency scaling mapping onto inverse scale change have profound impact on image analysis and scene generation. The emphasize them and address their importance in the following sections.

4.2 The Convolution Theorem

The convolution theorem relates frequency space to linear filtering. This is perhaps the most important theorem for linear system analysis. Simply, convolution in image space between a filter kernel and an input translates to multiplication of their Fourier transforms in frequency space. That is, given two functions $\phi(x)$ and $h(x)$ (*e.g.*, an input ϕ and a filter kernel h), and their Fourier transforms, $\Phi(\nu)$ and $H(\nu)$ the following relation holds:

$$\mathcal{F}(\phi(x) \otimes h(x)) = \Phi(\nu)H(\nu) \tag{23}$$

As with most relationships across the transform, the converse can also be easily generated.

$$\mathcal{F}^{-1}(\Phi(\nu) \otimes H(\nu)) = \phi(x)h(x) \tag{24}$$

Succintly, practitioners often say, *convolution in one domain is multiplication in the other domain.* Given this theorem, it is easy to conceive of why convolution is distributive over addition, symmetric, refexive, commutative, etc. It shares most of the properties of multiplication.

Linear filtering under these circumstances can now be considered in a different light. It can be changed from an integral with infinite extent in image space (a difficult continuous operation to implement with a digital computer) to a multiplication in frequency space, with only two Fourier transforms and one inverse Fourier transform to be calculated.

$$\phi(x) * h(x) = \mathcal{F}^{-1}(\mathcal{F}(\phi(x))\mathcal{F}(h(x))) \tag{25}$$

Depending on the size of both the filter kernel and in the input signal, casting the convolution problem in the frequency domain often makes difficult iterative problems more tractable. The Fourier transform itself is an integral with infinite extent. A discrete implementation that can be implemented on a digital computer is required to translate the Fourier transform from a useful mathematical abstraction to a practical tool.

4.3 The Discrete Fourier transform

How do we parley all of this frequency space theory into practical use? A typical digital image does not have infinite extent! Nor can it be sampled at infinitesimal rates, since it has already been sampled at a particular discrete sampling rate. The application of the Fourier transform to discretely sampled data results in the *Discrete Fourier Transform.*

Sinusoids are periodic, and for it to be computable, a decomposition of an input signal or image must also be considered periodic. To create a periodic signal, a discrete image with n uniformly spaced samples is *wrapped* so that samples 0 through n-1 index the data array exactly, but sample n "wraps" to 0. In fact, if n is the number of uniform samples, the effective index x_{eff} of a location x is, $x_{eff} = x \bmod n$. This can be applied to any dimension (*i.e.*, x, y, or z). This concept of a discrete image as a periodic signal can also be conceived as an infinite concatenation of the image to itself.

This repetition allows us to treat digital images as periodic signals. Moreover, it allows us to represent them as the sum of a finite (countable and finite) sinusoidal functions. The maximum representable wavelength is governed by dimensions and sampling rate of the input image. Consider a discrete input image F with n samples in each dimension. If the input image is considered to be a single period, we can map the domain of the input image to the interval from 0 to 2π (or from $-\pi$ to π). The dimension of each voxel then becomes $\frac{2\pi}{n}$.

The result is the discretization of the Fourier transform, with the sampled input function F transformed to a sampled frequency space representation representing the sum of n separate frequencies. The discrete inverse Fourier transform of the frequency space representation reproduces the original sampled data without loss of fidelity to the sampled signal. There may be a loss in the process of sampling. The implications to sampling become clear on an examination of the frequency space representation.

What remains is all the rest of sampling error, aliasing artifact, and digital fitler design. We cover some of these issues in the accompanying papers and provide references for extended reading on the subject.

5 Summary and Discussion

Filtering and frequency space are essential to the understanding of digital sampling theory. As we have seen in the convolution theorem, linear filtering and Fourier analysis are tightly linked, with profound implications in the understanding and manipulation of digital data. Aliasing and other sampling errors are directly related to the properties of the Fourier transform and the behavior of filters and input signals when discretized.

Frequency space arises everywhere, so a command of the basics is very important. The source of many volume images is in fact a manipulation of frequency space. The principals of image reconstruction in X-ray CT scanners and Magentic Resonance Imaging (MRI) are all based on uses of frequency space. Iterative methods, filtered back projection, etc. all have important roots in Fourier analysis. The resulting algorithms, even if they do not use a Fourier slice projection approach are often best understood in frequency space.

Basic control of these tools will help you get a handle on your data. Sampling, aliasing, level of detail, perspective texture correction (*e.g.*, "mip mapping"), and other common graphics techniques are only to be fully understood with through the window of frequency spaces.